Appendix C: little_r

C.1 Purpose

The purpose of the little_r program is to enhance the pressure-level, first-guess meteorological data with available observations contained within the analysis region. Any observation of wind speed and direction, temperature, dew point temperature or sea-level pressure, with an associated vertical location given as a pressure or a height, is acceptable. The input for the observations is formatted, ASCII text, which is intended to facilitate the "ease of use" consideration for new observation data sets.

The input first-guess data must be in a regridder format. No restrictions exist on the number of time periods to process, the number of vertical levels or the horizontal domain size. As long as memory is available, the program dynamically allocates the required space. For performance purposes, a namelist variable is used as input to allocate space for the number of observations, but modifying this value does not require recompilation. No parameter statements exist to define the anticipated domain size, as all of that information is input from the first-guess header.

The little_r program is able to perform an objective analysis for any subdomain in the regridder format. For each domain to be objectively analyzed, the little_r program must be separately run. A combination of analysis times may be mixed with gridded data without any observations, such as would occur for a real-time forecast. Only the input gridded data is used during the objective analysis, there is no vertical interpolation for additional levels or temporal interpolation for additional time periods.

The little_r program is written in FORTRAN 90, making extensive use of the new features in the FORTRAN language. This code has been successfully tested on several UNIX platforms. The little_r code is not one of the fully MesoUser supported programs: it has no traditional job deck and it cannot replace all of the features of RAWINS.

The little_r program has approximately 2500 lines of comments, with approximately 10,000 lines of executable code. This implies that directly turning to the source code for questions about the little_r program is encouraged and often rewarding.

C.2 Input Data

There are three input data streams to the little_r program: the namelist file, the observations and the first-guess fields. A template for the namelist file is provided and must be modified by the user to reflect the selected options. The observations are partitioned into separate files, one for each time period to be analyzed. The first-guess data is traditionally output from the previous modeling system program (regridder), though any pressure-level modeling system file is suitable for analysis by little_r. All time periods of the first-guess data are required to be in a single file.

C.2.1 Namelist

The little_r namelist is divided into nine logical records conforming to FORTRAN90 specifications. The namelist records are named "record1" through "record9", with each having a loosely related area of content. The namelist template file is "namelist.input" and is required to be in the current working directory of the little_r executable. The user must modify this file for each new objective analysis.

```
&record1
```

```
 start_date                       = '1990-03-13_00:00:00'
 end_date                         = '1990-03-13_00:00:00'
 interval                         = 21600/
```

The data in record1 concerns the temporal functioning of little_r.

Table C-1: RECORD1

Namelist Variable	Variable Type	Description
start_date	CHARACTER*19	YYYY-MM-DD_HH:mm:ss starting time to process
end_date	CHARACTER*19	YYYY-MM-DD_HH:mm:ss ending time to process
interval	INTEGER	time interval (s) between consecutive times to process

```
&record2
 fg_filename                  = '../data/kevin-fg'
 obs_filename                 = '../data/kevin-obs'/
```

The data in record2 concerns the names of the input files.

Table C-2: RECORD2

Namelist Variable	Variable Type	Description
fg_filename	CHARACTER	file name (may include directory information) of the first-guess fields, there is only a single name
obs_filename	CHARACTER	ile name(s) (may include directory information) of the observation files, one required for each time period to run through the objective analysis

```
&record3
 max_number_of_obs            = 10000
 fatal_if_exceed_max_obs      = .TRUE./
```

The data in the record3 concerns values that should not frequently need to be modified.

Table C-3: RECORD3

Namelist Variable	Variable Type	Description
mdate_size	INTEGER	size of an internal date, used for compatability with older Cray data sets

Table C-3: RECORD3

Namelist Variable	Variable Type	Description
max_number_of_obs	INTEGER	anticipated maximum number of reports per time period
fatal_if_exceed_max_obs	LOGICAL	T/F flag allows the user to decide the severity of not having enough space to store all of the available observations

```
&record4
 qc_test_error_max            = .TRUE.
 qc_test_buddy                = .TRUE.
 qc_test_vert_consistency     = .FALSE.
 qc_test_convective_adj       = .FALSE.
 max_error_t                  =  8
 max_error_uv                 = 10
 max_error_z                  = 16
 max_error_rh                 = 40
 max_error_p                  = 400
 max_buddy_t                  = 10
 max_buddy_uv                 = 12
 max_buddy_z                  = 16
 max_buddy_rh                 = 40
 max_buddy_p                  = 400
 buddy_weight                 = 1.0
 max_p_extend_t               = 1300
 max_p_extend_w               = 1300/
```

The data in record4 concerns the quality control options available. There are four specific tests that may be activatable by the user.

Table C-4: RECORD4 - QC Options

Namelist Variable	Variable Type	Description
qc_test_error_max	LOGICAL	difference between the first-guess and the observation
qc_test_buddy	LOGICAL	difference between a single observation and neighboring observations
qc_test_vert_consistency	LOGICAL	check for vertical spikes in temperature, dew point, wind speed and wind direction
qc_test_convective_adj	LOGICAL	removes any super-adiabatic lapse rate in a sounding by use of conservation of dry static energy

For the error maximum tests, there is a threshold for each variable. These values are scaled for time of day, surface characteristics and vertical level.

Table C-5: RECORD4 - Error Max Tolerances

Namelist Variable	Variable Type	Description
max_error_t	REAL	maximum allowable temperature difference (K)
max_error_uv	REAL	maximum allowable horizontal wind component difference (m/s)
max_error_z	REAL	not used
max_error_rh	REAL	maximum allowable relative humidity difference (%)
max_error_p	REAL	maximum allowable sea-level pressure difference (Pa)

For the buddy check test, there is a threshold for each variable. These values are similar to standard deviations.

Table C-6: RECORD4 - Buddy Check Tolerances

Namelist Variable	Variable Type	Description
max_buddy_t	REAL	maximum allowable temperature difference (K)
max_buddy_uv	REAL	maximum allowable horizontal wind component difference (m/s)
max_buddy_z	REAL	not used
max_buddy_rh	REAL	maximum allowable relative humidity difference (%)
max_buddy_p	REAL	maximum allowable sea-level pressure difference (Pa)
buddy_weight	REAL	value by which the buddy thresholds are scaled

For satellite and aircraft observations, data is occasionally horizontally spaced with only a single vertical level. The following two entries describe how far the user assumes that the data is valid in pressure space.

Table C-7: RECORD4 - Single Level Extension

Namelist Variable	Variable Type	Description
max_p_extend_t	REAL	pressure difference (Pa) through which a single temperature report may be extended
max_p_extend_w	REAL	pressure difference (Pa) through which a single wind report may be extended

```
&record5
 print_obs_files             = .TRUE.
 print_found_obs             = .TRUE.
 print_header                = .TRUE.
 print_analysis              = .TRUE.
 print_qc_vert               = .TRUE.
 print_qc_dry                = .TRUE.
 print_error_max             = .TRUE.
 print_buddy                 = .TRUE.
 print_oa                    = .TRUE./
```

The data in record5 concerns the enormous amount of print-out that can accompany the little_r program. These values are all logical flags that may be set to either TRUE or FALSE, where TRUE will generate output and FALSE will curtail additional print-out.

```
&record7
 use_first_guess             = .TRUE./
```

The data in record7 concerns early tests of the little_r program where the first-guess field was generated with the observations. Always use the first-guess data.

```
&record8
 smooth_type                 =  1
 smooth_sfc_wind             =  1
 smooth_sfc_temp             =  0
 smooth_sfc_rh               =  0
 smooth_sfc_slp              =  0
 smooth_upper_wind           =  0
 smooth_upper_temp           =  0
 smooth_upper_rh             =  0/
```

The data in record8 concerns the smoothing of the data after the objective analysis. The differences (observation minus first-guess) of the analyzed fields are smoothed, not the full fields.

Table C-8: RECORD8

Namelist Variable	Variable Type	Description
smooth_type	INTEGER	1 = five point stencil of 1-2-1 smoothing; 2 = smoother-desmoother
smooth_sfc_wind	INTEGER	number of smoothing passes for surface winds
smooth_sfc_temp	INTEGER	number of smoothing passes for surface temperature
smooth_sfc_rh	INTEGER	number of smoothing passes for surface relative humidity
smooth_sfc_slp	INTEGER	number of smoothing passes for sea-level pressure
smooth_upper_wind	INTEGER	number of smoothing passes for upper-air winds
smooth_upper_temp	INTEGER	number of smoothing passes for upper-air temperature
smooth_upper_rh	INTEGER	number of smoothing passes for upper-air relative humidity

```
&record9
 oa_type                        = 'MQD'
 mqd_minimum_num_obs            = 50
 mqd_maximum_num_obs            = 1000
 radius_influence               = 12
 oa_min_switch                  = .TRUE.
 oa_max_switch                  = .TRUE./
```

The data in record9 concerns the objective analysis options. There is no user control to select the various Cressman extensions for the radius of influence (circular, elliptical or banana).

Table C-9: RECORD9

Namelist Variable	Variable Type	Description
oa_type	CHARACTER	"`MQD`" for multiquadric; "`Cressman`" for the Cressman-type scheme, this string is case dependent
mqd_minimum_num_obs	INTEGER	minimum number of observations for MQD
mqd_maximum_num_obs	INTEGER	maximum number of observations for MQD
radius_influence	INTEGER	radius of influence in grid units for Cressman scheme

Table C-9: RECORD9

Namelist Variable	Variable Type	Description
oa_min_switch	LOGICAL	T = switch to Cressman if too few observations for MQD
oa_max_switch	LOGICAL	T = switch to Cressman if too many observations for MQD

C.2.2 Observations

The observations that are input to little_r come in formatted, ASCII text files. For sorting performance concerns, each time period is stored in a separate file. The little_r program is able to combine reports from the same station (such as mandatory levels and significant levels). Duplicate observations are removed. The observations are vertically interpolated to the same levels as the input first-guess data, where a maximum interpolation depth (15000 Pa) is not exceeded. A sample program (upa.f in the util directory of the little_r tar file) provides the exact format as defined by a FORTRAN program. The bogus data is included in the same file as the regular data with a LOGICAL switch used to differentiate the two sources.

The user is responsible for reordering the observations into the required format for little_r input. Internal codes exist to permit the use of the archived NCAR surface and upper-air reports (as have been used in RAWINS) and the raw observations (not netCDF), which are available in real-time on the UNIDATA circuit. The file upa.f, which puts the observations in the format expected by little_r, assumes that the user is already able to ingest the native observation data.

Though it is recommended that separate time periods be placed in separate files, this is not strictly a requirement. The little_r program opens, reads and then closes the sequential set of observation file names listed in the namelist.input file in synchronization with the number of the time period being processed. During the processing of the first time period, as computed form the starting date, the first observation file is used. During the processing of the second time period, the second observation file is used, and so on. For all requested times to process that extend beyond the number of observation files, the first-guess data is reformatted into a RAWINS structure and output with no attempt at enhancement.

C.2.3 Historical Observation Access

A Cray job deck has been written to allow users with NCAR Cray access to utilize the traditional MSS archives previously available to RAWINS. The deck, fetch.deck, pulls in the data for a requested time period, converts it to a format acceptable for little_r, and then rcp's the data to a specified local machine.

```
# QSUB -r fetch                         # request name
# QSUB -q prem                          # job queue class
# QSUB -o fetch.SEQN
# QSUB -eo                              # stdout and stderr together
# QSUB -lM 10Mw                         # maximum memory
# QSUB -lT 3000                         # time limit
# QSUB                                  # no more qsub commands
```

```
ja

#               ****************************************
#               ****** fetch interactive/batch C shell *****
#               *******         V2 System          ******
#               *******          f90 only          ******
#               ****************************************

#    Note: when running this deck interactively, it assumes
#    the source files are local and un-tared.

#    This shell fetches ADP data from the NCAR MSS system and
#    converts it into a format suitable for the little_r
#    program.  The data are sent back to the local machine and
#    stored on the NCAR MSS.

#    This should be the user's case or experiment (used in MSS name).
#    This is where the data will be stored on the MSS.

set ExpName   = MM5V2/TEST    # MSS path name for output
set RetPd     = 365           # MSS retention period in days

#    Location where data are to be returned on local machine.

set Host     = username@host.domain:/usr/tmp/username
set Host     = headache2.mmm:/headache2/gill/DATA

#    The only user inputs to the fetch program are the beginning
#    and ending dates of the observations, and a bounding box for the
#    observation search.  These dates are given in YYYYMMDDHH.  The
#    ADP data are global, and include the surface observations and
#    upper-air soundings.  A restrictive bounding box (where
#    possible) reduces the cost substantially.

set starting_date = 1993031300
set ending_date   = 1993031400

set lon_e         =  180
set lon_w         = -180
set lat_s         =  -90
set lat_n         =   90

set lon_e         =  -50
set lon_w         = -120
set lat_s         =   20
set lat_n         =   70

#    Is there a personal copy of the source?

set UseMySource = no

#########################################################
#########                                       #########
#########       END OF USER  MODIFICATIONS      #########
#########                                       #########
#########################################################
```

The user need only specify the bounding dates, since the program looks up the necessary files based on when various data archives were active. Because of the really poor performace that Cray machines have with regard to tar and gzip, you significantly reduce the CPU and elapsed time by providing a gross latitude-longitude box within which your observations may be found. The fetch.deck file is available on the NCAR anonymous ftp site (ftp.ucar.edu), under `mesouser/newprogs/fetch.tar.gz`.

C.2.4 First-Guess

The first-guess data may be any pressure-level data format produced by the MM5 system (regridder format, little_r format or interpolated model output format). The little_r program accepts data in any of these formats, but always generates output in the little_r format. Where each time period is a separate file for the observations, the first-guess data are concatenated (typical of the MM5 system gridded data formats) so that all of the time periods are contained within a single file.

The little_r program is able to process gridded data in any of the three traditional MM5 system projections: Mercator, Lambert conformal and polar stereographic. If the gridded field is the most coarse domain (domain ID #1 in the header), then the input first-guess data may use an expanded domain to allow observations outside of the forecast domain to affect the boundary analysis. The expanded domain is cut down to the forecast domain at the end of the little_r processing of the time period. Since the size of the internal gridded arrays are dynamically allocated based upon header information, no restrictions exist for the horizontal or vertical extent of the data: different cases may use the same little_r executable.

C.3 Output Data

The little_r program generates several ASCII text files to detail the actions taken on observations through a time cycle of the program (sorting, error checking, quality control flags, vertical interpolation). In support of users wishing to plot the observations used for each variable (at each level, at each time), a file is created with this information. Primarily, the ASCII text files are for consumption by the developers for diagnostic purposes. The main output of the little_r program is the gridded, pressure-level data set to be passed to the INTERP program. There is no surface FDDA file created by little_r.

In each of the files listed below, the text "_YYYY-MM-DD_HH:mm:ss.tttt" allows each time period that is processed by little_r to output a separate file. The only unusual information in the date string is the final four letters "tttt" which is the decimal time to ten thousandths of a second.

C.3.1 result_out_YYYY-MM-DD_HH:mm:ss.tttt

This file contains a listing of all of the observations available for use by the little_r program. The observations have been sorted and the duplicates have been removed. Observations outside of the analysis region have been removed. Observations with no information have been removed. All reports for each separate location (different levels but at the same time) have been combined to form a single report. Interspersed with the output data are lines to separate each report. This file contains discarded reports.

C.3.2 useful_out_YYYY-MM-DD_HH:mm:ss.tttt

This file contains a listing of all of the observations available for use by the little_r program. The observations have been sorted and the duplicates have been removed. Observations outside of the analysis region have been removed. Observations with no information have been removed. All reports for each separate location (different levels but at the same time) have been combined to form a single report. Data which has had the "discard" flag internally set (data which will not be sent to the quality control or objective analysis portions of the code) are not listed in this output. No additional lines are introduced to the output, allowing this file to be reused as an observation input file.

C.3.3 discard_out_YYYY-MM-DD_HH:mm:ss.tttt

This file only contains a listing of each of the discarded reports. This is a good place to begin to search to determine why an observation didn't make it into the analysis. This file also has additional lines interspersed within the output to separate each report.

C.3.4 qc_out_YYYY-MM-DD_HH:mm:ss.tttt

The information contained in the qc_out file is similar to the useful_out. The data has gone through a more expensive test to determine if the report is within the analysis region, and the data have been given various quality control flags. Unless a blatant error in the data is detected (such as a negative sea-level pressure), the observation data are not tyically modified, but only assigned quality control flags. Any data failing the error maximum or buddy check tests are not used in the objective analysis.

C.3.5 obs_used_for_oa_out_YYYY-MM-DD_HH:mm:ss.tttt

This file lists data by variable and by level, where each observation that has gone into the objective analysis is grouped with all of the associated observations for plotting or some other diagnostic purpose. The first line of this file is the necessary FORTRAN format required to input the data. There are titles over the data columns to aid in the information identification. Below are a few lines from a typical file.

```
( 3x,a8,3x,i6,3x,i5,3x,a8,3x,2(g13.6,3x),2(f7.2,3x),i7 )
Number of Observations 00001214
Variable   Press     Obs      Station           Obs           Obs-1st      X          Y        QC
Name       Level    Number     ID              Value           Guess     Location  Location  Value
U           1001        1   CYYT          6.39806         4.67690      161.51    122.96         0
U           1001        2   CWRA          2.04794        0.891641      162.04    120.03         0
U           1001        3   CWVA          1.30433        -1.80660      159.54    125.52         0
U           1001        4   CWAR          1.20569         1.07567      159.53    121.07         0
U           1001        5   CYQX         0.470500        -2.10306      156.58    125.17         0
U           1001        6   CWDO         0.789376        -3.03728      155.34    127.02         0
U           1001        7   CWDS         0.846182         2.14755      157.37    118.95         0
```

C.3.6 *first_guess*_+analysis

By default, the name of the final analysis file uses the first-guess name (given in the namelist.input file) as a prefix and adds the string "`_+analysis`", including the directory information. Users must therefore have write permission in the directory where the first-guess data is located. This output file is the little_r format file. If every time period of the first-guess data was processed and the original data set was not expanded, the final analysis file is the same size as the first-guess file. No additional levels, time periods or fields are added to the final analysis data.

C.4 Quality Control Techniques

There are several levels of quality control for the observations that are employed. All of the methods refer only to the specific time that is being processed, so that no historical or time-series information is used to determine the representativeness of an observation. Only data that has not been discarded (an official status inside of little_r for a report) and is not bogus data is available for quality control procedures. No matter how heinous the value, bogus data is not required to pass through any quality control procedure. For tests that require a comparison between the first-guess field and an observation, the comparison is made at the observation site through use of the surrounding four first-guess points with a bi-linear interpolation.

The quality comtrol values are of the form 2^n, where n takes on positive integer values. This allows the various quality control flags to be additive yet permitting the decomposition of the total sum into constituent components. Following are the current quality control flags that are applied to observations.

```
pressure interpolated from first-guess height          = 2 **  1 =      2
temperature and dew point both = 0                     = 2 **  4 =     16
wind speed and direction both = 0                      = 2 **  5 =     32
wind speed negative                                    = 2 **  6 =     64
wind direction < 0 or > 360                            = 2 **  7 =    128
level vertically interpolated                          = 2 **  8 =    256
value vertically extrapolated from single level        = 2 **  9 =    512
sign of temperature reversed                           = 2 ** 10 =   1024
superadiabatic level detected                          = 2 ** 11 =   2048
vertical spike in wind speed or direction              = 2 ** 12 =   4096
convective adjustment applied to temperature field     = 2 ** 13 =   8192
no neighboring observations for buddy check            = 2 ** 14 =  16384
fails error maximum test                               = 2 ** 15 =  32768
fails buddy test                                       = 2 ** 16 =  65536
observation outside of domain detected by QC           = 2 ** 17 = 131072
```

C.4.1 Gross Error Checks

The first line of defense against suspect data is on input. Glaring errors such as negative values for positive definite fields, and other gross bound checks for wind, temperature and pressure are applied. Observations that exceed the gross bound values are reset to a flag value for missing data.

C.4.2 Error Maximum Checks

A comparison is made between each observation and the value interpolated from the first-guess data at the point of the observation. Prior to this comparison, the observation either existed at the vertical level of the first-guess field or had been vertically interpolated to that level. No comparison is possible for levels of the observation for which no first-guess field is available (such as at most of the significant levels, though the significant levels are used in the vertical interpolation to the first-guess analysis levels).

In addition to the specified maximum difference values that were defined in the namelist file, scale factors modify the error tolerances further (larger tolerances allow in more data). The horizontal components of wind allow the error to increase by 25% between 1000 hPa and 500 hPa; above 500 hPa the error tolerance increases by 50%. For surface temperature, the tolerance increases by 50% if the approximate local time is between 1100 and 2000 hours, and is increased by 25% at all other times. For temperature observations between 1000 hPa and 700 hPa, the tolerance is reduced by 15%; for temperatures above 700 hPa, the tolerance is reduced by 35%. No scale factors are applied to sea-level pressure or relative humidity.

C.4.3 Buddy Checks

The buddy check routine performs error checks by comparing the difference between the first-guess field and observations at the observing locations with the averaged difference at the nearby stations within a given distance. The distance for the buddy observations is given as 500 km for surface observations, 300 km for observations between 1000 hPa and 200 hPa, and 650 km for observations above 200 hPa.

If the difference value at the observation point differs from the average of the nearby difference values by more than a certain maximum (dependent on variable types, pressure levels, and the input value parameter from the namelist), the observation is flagged as suspect, and the "fails buddy check" flag is added to the quality control value.

Similar to the error maximum criteria, scale factors are applied to the single input value from the namelist for each variable. For the horizontal components of the wind, all levels at 400 hPa and above have the error tolerance scaled by 1.7, between 700 hPa and 400 hPa are scaled by 1.4, from 850 hPa through 700 hPa scaled by 1.1, and all levels below 850 hPa are unity scaled. For temperature, all of the error tolerances for levels at and above 400 hPa are scaled by 0.6, all levels between 700 hPa and 400 hPa are scaled by 0.8, and all levels below 700 hPa are unity scaled. There are no scalings for the error tolerances of the sea-level pressure or relative humidity.

C.4.4 Vertical Consistency Checks

Other than the previously mentioned tests that check for large errors based on reasonable bound checks, this is one of the two checks that will modify values in the report. A minimum of three vertical levels are required. Tests are conducted for superadiabatic lapse rates and for very strong inversions. If the temperature profile can be "improved" by switching the sign of the temperature (C), this is performed. Any modifications to the temperature affect the relative humidity by keeping the depression invariant.

The horizontal components of the wind may also be adjusted by the tests conducted on the wind speed and direction. For each consecutive three layers, second order vertical derivatives are computed for both speed and direction. The absolute value of these differences and general speed increase with height are considerations for data acceptance. No attempt is made to correct the wind speed or direction. If the data are acceptable, there are no modifications; if the data is suspect, the wind information is set to the flag value for missing data.

The bounds used in these tests seem fairly arbitrary and therefore the vertical consistency checks are not routinely activated.

C.4.5 Convective Adjustment Checks

This routine detects locations of superadiabatic lapse rates and corrects the temperature through the conservation of dry static energy (g*Z + Cp*T). When a superadiabatic layer is found, an upward and downward sweep from that point through the sounding is taken to determine the bounding extremities of the correction. For each temperature correction, the associated dew point and relative humidity are recomputed by assuming the depression is invariant. This is the second quality control check that modifes data.

This test is not typically activated. For example, there are occurrences where surface superadiabatic layers are both physically realistic and desirable in the analysis.

C.5 Objective Analysis Techniques

There are two available choices for the objective analysis method: a Cressman-type scheme and a multiquadric method. The Cressman scheme uses an anisotropic radius of influence (adjusted by wind speed, direction and curvature) to modify grid points in the vicinity of each observation. The multiquadric scheme relates the influence of each observation to every grid point in the domain based upon distance and similarly correlates each observation to every other observation through an inverse distance metric.

The data that are objectively analyzed for both the Cressman and multiquadric schemes are the interpolated differences at each of the observation points. After a possible smoothing operation on the objectively analyzed field, this perturbation field is added back to the original first-guess field to generate the final analysis. Each variable (u, v, temperature, relative humidity, sea-level pressure), at each level is analyzed independently.

If users choose the Cressman scheme, all of the objective analysis is handled through the Cressman routines. If the user specifies the multiquadric scheme, and either the minimum or maximum flag for the number of observations, the multiquadric scheme may be used at a particular level and the Cressman scheme used at a different level for the same variable when the number of available observations falls outside the prescribed range.

C.5.1 Cressman-type Scheme

The Cressman scheme in little_r allows for a circular radius of influence, an elliptical radius of influence (in the direction of strong flow) and a curved, elliptically shaped (banana) radius of influence (in areas of strong curvature). These options are automatically implemented based on a critical wind speed and an arbitrarily defined radius of curvature for the first-guess wind field.

The critical wind speed is 5 m/s at 1000 hPa, linearly increases to 15 m/s at 500 hPa and remains at 15 m/s above 500 hPa. For all locations where the first-guess is slower than this critical wind speed, the radius of influence is circular. If the radius of curvature of the first-guess gridded field is less than three times the radius of influence (strongly curved flow) and the first-guess wind speed is above the critical minimum, then the radius of influence is banana shaped. For wind speeds in excess of the critical wind speed, but with a larger radius of curvature, the radius of influence is elliptically shaped.

The user has no access to namelist switches to modify these default fall-back options or their activating circumstances. Additionally, the weighting term is squared in the numerator, which tends to remove the signature “circles” around temperature and relative humidity observations. This implementation allows multiple scans of the Cressman scheme, where each iteration uses the previous output as the new first-guess. Typically, the successive scans employ a telescoping radius of influence to increasingly emphasize the localization of each observation.

C.5.2 Multiquadric Scheme

The multiquadric technique involves inverting a symmetric observation correlation matrix, where the correlation is based upon distance (how far away is every observation away from every other observation). The diagonal elements are based on information, such as variable type and number of observations. This inverse is premultiplied by a rectangular matrix which has as elements the distance of every observation to every grid point, and postmultiplied by the perturbation observation vector. This provides an influence array for each grid point by each observation.

C.6 little_r Didn't Work, What Went Wrong?

Nothing, this is a trick question. The little_r program has never failed to run, and the program always produces glorious results. Users are occasionally surprised at the "features" that are in little_r that appear to closely resemble bugs.

The little_r program is able to have extensive print-out information. Most of the time, these are able to capture the problem that the program is experiencing. The first place to look is to do a UNIX "tail" of the print-out.

To find out how many observation reports are available to the program, search for "`Number`" in the output:

```
Number of observations successfully ingested:          1587.
Number of empty observations discarded:                 100.
Number of observations discarded outside of domain:     191.
```

To see how many observations went into the objective analysis for each variable at each level, search for "`for OA,` ". The output is arranged by variable for each level. The first five searches for "`for OA,` " will be for the surface u, surface v, surface temperature, surface realtive humidity and sea-level pressure, respectively. There after, each of the following searches will be for the four variables u, v, temperature and relative humidity, respectively at a particular pressure level (starting at 1000 hPa and continuing up through the top of the analysis). Below is an example of this text.

```
Of the  1587 observations found in the observation input file, for OA, only  1214
of them are available inside the domain. This field is U        for pressure
level 1001 hPa (1001 hPa is defined as the surface).

Variable   Press    Obs     Station         Obs          Obs-1st      X         Y        QC
Name      Level    Number    ID            Value          Guess    Location  Location  Value
U          1001       1   CYYT         6.39806        4.67690       161.51    122.96       0
U          1001       2   CWRA         2.04794       0.891641       162.04    120.03       0
U          1001       3   CWVA         1.30433       -1.80660       159.54    125.52       0
U          1001       4   CWAR         1.20569        1.07567       159.53    121.07       0
 ...
U          1001    1210   CYMO        0.997769       -5.96655       106.53    120.47       0
U          1001    1211   CYYR         1.04954     -0.342371E-01    141.84    133.59       0
U          1001    1212   CWSE         9.51595        8.90193        47.24   135.75    16384
U          1001    1213   PANT       0.907688E-01    -2.76653        21.71    153.36       0
U          1001    1214   CYYQ       -0.163728       -1.50877        83.65    143.40       0
```

The little_r program must have the namelist.input file in the working directory, and the input files listed within the namelist file must exist (though the observation file can be empty). The requested dates in the namelist file must exist in the first-guess data set.

Frequent troubles extend from trying to write the observations in a format that the little_r program expects to find, which is why the sample program (upa.f) has been provided. All units are SI for the observation data: wind (m/s), temperature (K), pressure (Pa), height (m). Use separate files for separate observation times that are associated with the different final analysis times to process.

A large disparity in the data distribution for the observations (such as would happen if half of the domain is continental with a dense data network and the other half of the domain is oceanic with few observations) increases the chances for a poor analysis if the multiqudric method is chosen. The multiquadric method is two-dimensional, but the effective radius of influence is the entire analysis domain.

The radius of influence for the Cressman scheme is given in grid point units, where each grid distance is approximately equal to the grid point separation defined in the data set header. Double check the use of the units.

C.7 How to Run little_r

The little_r program does not have a standard MM5 system job deck C shell. The job is controlled through the use of the namelist file (namelist.input). Following are the required steps to run the little_r program. It is assumed that the user has a first-guess file (traditionally this is a regridder formatted file) and observation files (such as from the fetch.deck program) for each of the time periods to be processed through the objective analysis.

C.7.1 Get the little_r source code

The NCAR MesoUser maintains the little_r compressed tar file on the NCAR anonymous ftp site.

```
# ftp ftp.ucar.edu
Name: anonymous
Password: your_complete_email_address
ftp> cd mesouser/newprogs
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 18068
drwxrwsr-x   2 9001     24           512 Nov 23 20:34 .
drwxrwsr-x  14 9001     24           512 Aug 14 15:08 ..
-rw-rw-r--   1 9001     24         16106 Nov 23 20:35 getsfcobs.tar.gz
-rw-rw-r--   1 9001     24        118344 Dec  8 18:36 little_r.tar.gz
-rw-rw-r--   1 9001     24       8916943 Nov 18 17:47 little_r_data.tar.gz
-rw-rw-r--   1 9001     24        163683 Dec 11 22:46 regrid.tar.gz
226 Transfer complete.
ftp> bin
ftp> get little_r.tar.gz
ftp> quit
```

The compressed tar file contains the little_r source code and utility files. The data is first uncompressed and then untarred. This will make a “little_r” directory.

```
# gunzip little_r.tar.gz
# tar -xf little_r.tar
```

Get into the little_r directory and we are ready to get rolling.

```
# cd little_r
```

C.7.2 Set Up the Compiler Directives, Generate the Executable

Users will find several README files in the Doc directory. References are made in these README files to test data sets that exist on the NCAR anonymous ftp site (that is the big file little_r_data.tar.gz above). If things are not working with your data, it is probably worthwhile to download this file.

All of the little_r source code is contained in the src directory. Note that FORTRAN90 is a requirement. The program is run from the main directory. For users running on a Cray, SGI, DEC, IBM or HP, the Makefile is already set up to build the executable. For a Compaq machine, the user types:

```
# make dec
```

Please send any compiler/loader option suggestions back to MesoUser (*mesouser@ucar.edu*), particularly if your machine is not among the available choices in the Makefile.

This should be sufficient to build the executable "little_r". Note that there is no need to define the domain configuration prior to making the executable. All of its required memory is allocated through a maximal value in the namelist and through the use of the header information from the first-guess data.

C.7.3 Run the Program

Information given previously in this appendix describes how to set up the namelist.input file, and a README file also exists in the directory above src. The namelist file must be in the current working directory of the little_r executable. The observation files and the first-guess data that are listed in the namelist.input file must exist (directory information may be given in the name string). The little_r program is able to generate a large amount of print information, so it is recommended that the user capture it in a file.

```
# little_r > output
```

C.7.4 Did It Run Successfully?

The last line in the little_r output should be

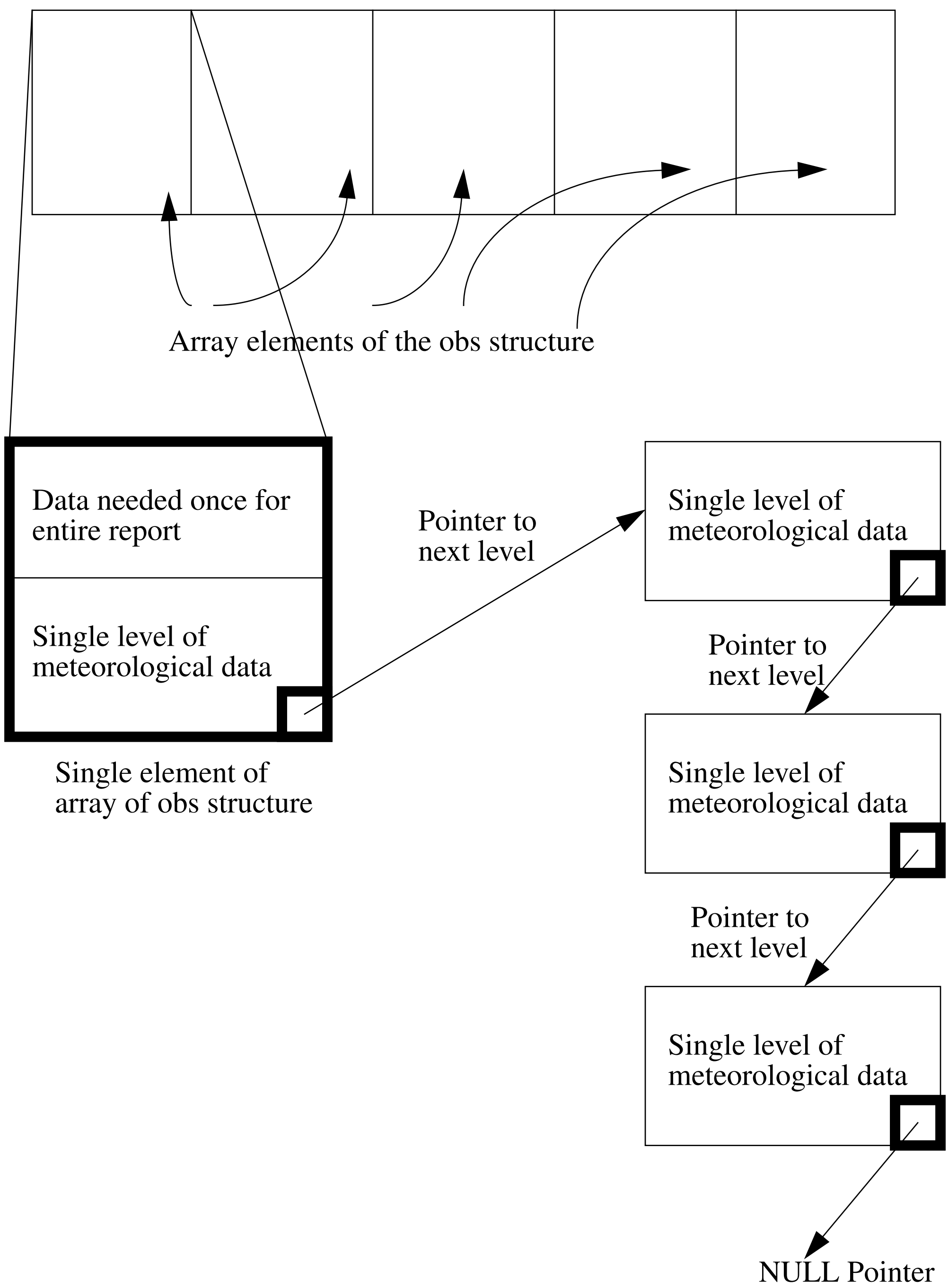
```
STOP 99999
```

if everything went OK, though this is necessary for a successful completion, it is not sufficient. If all of the time periods in the first-guess data were processed by little_r, and if the input data was not using the expanded option, and if the input data was either in the regridder or little_r format (not interpolated MM5 output), then the final analysis file should be identical in size to the first-guess data file.

C.8 Structure and Methods

The little_r program uses a FORTRAN90 data structure throughout the code to store and access the individual reports. This is an array of a user defined type. The length of the array is set up from the namelist.input file. Most of the actual storage required for observation data is tied up in the information at each new level. Typically, it was the number of these levels that was hard coded in FORTRAN77 that wasted significant amounts of memory. In little_r, the user defined type has pointers to "yet another level of data". This linked list of data on the same level for a single report is dynamically allocated, minimizing the required machine memory.

C.8.1 Observation Storage



C.8.2 Observation Structure

The observation file for the little_r program is an ASCII text file, though with the long record length, it would be difficult to edit directly. The observation file usually is for a specific time for performance reasons, however, the user may include all of the observations for every time period in a single observation file. In such a case, to have multiple analysis times, the namelist would have the observation file name replicated for each separate time period.

The units used in the observation file are MKS: pressure and sea-level pressure are in Pa, geopotential height and elevation are in m, temperature and dew point are in K, wind speed and the horizontal wind components are in m/s, and wind direction is in degrees. The latitude and longitude are in decimal degrees, where positive refers to north and east, respectively.

The observation data do not need to be sorted. The program handles combining pieces of reports (such as mandatory and significant levels), filtering data by time, and removing duplicate information.

There are three different types of records in the observation file: rpt_format, meas_format and end format. The first line is a rpt_format, which contains the date, location, observation name, and some surface information. There follows at least one meas_format record, though usually there are multiple records. Each meas_format record provides information for all of the traditional upper-air fields at a particular level. Each meas_format record is for a new vertical level. The end of the report is signaled by a end_format record. These combined records constitute an entire input report, though as mentioned, several input reports may be combined by little_r. Below are the definitions of these format strings.

```
CHARACTER ( LEN = 120 ) , PARAMETER :: rpt_format =  &
             ' ( 2f20.5 , 2a40 , ' &              ! format for location_type
          // ' 2a40 , 1f20.5 , 5i10 , 3L10 , ' & ! format for source_info
          // ' 2i10 , a20 , ' &                   ! fmt for valid_time
          // ' 13( f13.5 , i7 ) ) '               ! fmt for 'terrestrial'

CHARACTER ( LEN = 120 ) , PARAMETER :: meas_format = &
             ' ( 10( f13.5 , i7 ) ) '             ! fmt for measurement rcd

CHARACTER ( LEN = 120 ) , PARAMETER :: end_format = &
             ' ( 3 ( i7 ) ) '                     ! fmt for end record
```

Following is the source code used to build the user defined type that holds a single meteorological report. An array of the user defined type holds all of the available observations for a single time period of processing. At the end of the time period, the entire linked list is de-allocated.

```
!---------------------------------------------------------------------------

   TYPE location_type

      !  The fields in this record uniquely identify the source of the
      !  data, so that duplicates of data can be merged or discarded.
      !  The horizontal location of this report (assumed constant, even
      !  for balloon ascent) is geven by the lat/lon of the site.

      REAL                    :: latitude  , &   ! latitude (+ degrees east)
                                 longitude       ! longitude (+ degrees north)

      CHARACTER ( LEN = 40 ) :: id , &           ! 5 digit identifier,
                                               ! consisting of a 2 digit block
                                                 ! number and a 3 digit
                                                 ! identifier (for soundings)
                                                 ! for WMO sttn; non digit
                                                 ! for other sources
                                 name            ! The name corresponds to
```

```
                                             ! the id (is obtained from id
                                             ! in the program that is
                                             ! source of data
   END TYPE location_type

!---------------------------------------------------------------------------

   TYPE source_info

      CHARACTER ( LEN = 40 ) :: platform , &    ! description of the
                                                ! measurement device
                                source          ! GTS data, NCAR ADP files,
                                                ! bogus information, etc
      REAL                   :: elevation       ! station elevation

      !  During the decoding process, how many errors (non conforming
      !  codes) were encountered, and how many warnings (this is a subjective
      !  call based on repeated incorrect -- but probably not garbled --
      !  GTS codes).  If a bulletin is completely garbled, the logical
      !  flag to not consider this report is set.

      INTEGER              :: num_vld_fld , & ! number of valid fields in the
                                              ! entire report; used as the
                                              ! first tie-breaker in deciding
                                              ! which conflicting data items
                                              ! to keep if have duplicate rpts
                              num_error , &   ! number of errors
                                              ! encountered during the
                                              ! decoding process
                              num_warning , & ! number of warnings
                                              ! encountered during the
                                              ! decoding process
                              seq_num , &     ! sequence numbers that tell
                                              ! which of 2 reports is more
                                              ! recent.
                              num_dups        ! number of duplicates found of
                                              ! this observation
      LOGICAL              :: is_sound        ! is-a-sounding tells whether
                                              ! the observation possibly has
                                              ! multiple levels vs having
                                              ! only one level for srfc ob.
      LOGICAL              :: bogus           ! T/F if this is a bogus
                                              ! observation
      LOGICAL              :: discard         ! Tells whether this observation
                                              ! has been found to be a dup
                                              ! AND has been discarded or
                                              ! merged.
   END TYPE source_info

!--------------------------------------------------------------------------

   TYPE field

      !  Defines a data type consisting of a paired data value (real) with a
      !  quality control flag that holds a binary-coded combination of error
      !  codes; the codes  identify possible problems with the data.

      REAL                   :: data
      INTEGER                :: qc              !  Quality control flags
```

```
                                                   !  that are 0 if data is
                                                   !  good, or different
                                                   !  integers depending upon
                                                   !  what error(s) occured
   END TYPE field

!--------------------------------------------------------------------------

   TYPE terrestrial

      !  The data that will occur, at most, once during a report is
      !  listed here.  These are typically terrestrial measured values.  The
      !  surface met fields are stored in a separate TYPE, to allow a
      !  POINTER to the next level (if one exists).  This needs to be a
      !  separate TYPE so as to allow a POINTER to it

      TYPE ( field )          :: slp        , &    ! sea level pressure
                                 ref_pres   , &    ! reference pres level for
                                                   ! the thickness
                                 ground_t   , &    ! ground temperature
                                 sst        , &    ! sea surface temperature
                                 psfc       , &    ! surface pressure
                                 precip     , &    ! precipitation accumulation
                                 t_max      , &    ! daily temperature max
                                 t_min      , &    ! daily temperature min
                                 t_min_night , &   ! min overnight temperature
                                 p_tend03   , &    ! pressure tendency in 3hr
                                 p_tend24   , &    ! pressure tendency in 24hr
                                 cloud_cvr  , &    ! total cloud cover (oktas)
                                 ceiling           ! height of lowest cloud base
   END TYPE terrestrial

!--------------------------------------------------------------------------

   TYPE time_info

     !  GTS report time: the valid time of the report.  The largest INTEGER values
      !  require only 8 digits, so that this should function properly with
      !  32-bit INTEGERS.

      INTEGER                 :: sut       , &     ! number of seconds since 1 Jan
                                                   ! 0000 UTC 1970
                                 julian            ! Julian day
      CHARACTER ( LEN = 14 )     date_char         ! CCYYMMDDHHmmss date

   END TYPE time_info

!---------------------------------------------------------------------------

   TYPE meas_data

      !  The met data involved with this program is defined in this TYPE.  The
      !  standard state variables (wind, temperature, moisture, with pressure
      !  and/or height to fix the vertical location) are stored here.  For
      !  single level observations, only one of these records is used per
      !  observation.   For multi-level reports, a linked list of these
      !  measurement TYPEs is generated.

      TYPE ( field )          :: pressure    , & ! pressure of observation
                                 height      , & ! height (above sea level)
```

```
                                        temperature , & !
                                        dew_point   , & !
                                        speed       , & !
                                        direction   , & !
                                        u           , & ! u and v components of wind
                                        v           , & ! are derived from spd and dir
                                        rh          , & !
                                        thickness       !

   END TYPE meas_data

!---------------------------------------------------------------------------

   TYPE measurement

      TYPE ( meas_data )                :: meas  ! contains data and qc code
      TYPE ( measurement ) ,  POINTER   :: next  ! the met info is handled
                                                 ! as a linked list of the
                                                 ! measurement type

   END TYPE measurement

!--------------------------------------------------------------------------

   TYPE report
                                                ! this is the entire report
      TYPE ( location_type ) :: location        ! for a single time, from a
      TYPE ( source_info )   :: info            ! single reporting platform,
      TYPE ( time_info )     :: valid_time      ! a sounding, surface, buoy,
      TYPE ( terrestrial )   :: ground          ! aircraft or ship report
      TYPE ( measurement ) , &
               POINTER       :: surface

   END TYPE report
```

The first read statement of a report fills in the location (location_type type), info (source_info type), valid_time (time_info type), and ground information (terrestial type). The first character string is for the stations ID. The first 5 characters should be the 2-digit block code and the 3-digit identifier. The remainder of this character string may contain the name of the station (used for sorting and station uniqueness tests). The second character string is to provide information about the source of the observations, but is not used in the processing. The third character string needs to have one of a few predefined labels: such as "FM-35 TEMP" or "FM-12 SYNOP", for upper air or surface data, respectively. A few other WMO values are acceptable. As given in the previous pages of defined types, options for bogus data and upper-air flags are defined in the intitial rpt_meas record. The date is in the YYYYMMDDHHmmss format. Below is the read statement which uses the previously defined data types.

```
      READ ( file_num , FMT = rpt_format ) &
             obs(obs_num)%location , &
             obs(obs_num)%info , &
             obs(obs_num)%valid_time , &
             obs(obs_num)%ground
```

Note that for the ground data, there is information for the 13 fields, where each field is comprised of an observation value and a quality control flag. On input, the QC flags are defined to be zero. Only the sea-level pressure field is used by little_r, the rest of the measured fields are ignored.

Missing data is defined as -888888.0.

The majority of the meteorological data is contained on the multiple lines of the meas-format records. Each record has observation and QC information for ten variables (see the meas_data type). On input, pressure or height must be defined. If pressure is missing, it is generated based on the observation height and the first guess height field. The temperature, dew point, wind speed and direction are also input. The relative humidity, and u and v components of wind are diagnosed and should not be input. Missing data is defined as -888888.0. Below is the read statement used to pull in a single vertical level of the meteorological data.

```
READ ( file_num , FMT = meas_format ) surface%meas
```

To signal the end of the report, a final meas-format record is written with the pressure and height defined as -777777.0, the temperature defined as the number of number of levels, and the rest of the fields set to missing (-888888.0).

When the program reads the trailing sentinel flags denoting the end of the data, the last read statement is for the end_format. This information is used to aid in the decision of which observation value to use should an option be available. In the program that decodes the raw WMO data, these three integers are the total number of valid fields, the number of errors in the decoding (non-conformance with the standard) and the number of warnings.

```
READ ( file_num , FMT = end_format ) &
       obs(obs_num)%info%num_vld_fld , &
       obs(obs_num)%info%num_error , &
       obs(obs_num)%info%num_warning
```

For practical considerations, users may set these values to the total number of fields x levels, 0 and 0, respectively.

C.8.3 Program Processing Order

The little_r program is initiated from the main program (main). There is a time loop over the driver routine (which handles all of the processing for a single time period). After the time loop completes, the error handler prints out the information/error history, and the programs stops.

There are several file naming conventions used in the little_r program that aid in understanding the calling structure:

1. Compilable files end with “.F”
2. Module files use “_module.F”
3. Programs that interface exclusively with a module use the prefix “proc_”.
4. The files “proc_oa.F” and “oa_module.F”, for example, are related. The subroutine “proc_oa.F” calls subroutines in the module “oa_module.F”.
5. All interface block files use the extension “.int”.
6. All include files use the extension “.inc”. Only the map information is shared through a common block. All other include files are for the purposes of defining data types consistently in several routines.

main

proc_namelist - namelist exist, read, fill namelist structure

proc_header - open, read, compatible with namelist

loop over time

driver

proc_get_info_header - fill all local useful constants from header, error checking compared to namelist, print available values

proc_first_guess - ingest first-guess data, define window, fill arrays

proc_obs_sort - ingest, sort, rotate winds, decide if inside domain, vertically interpolate obs to required levels

proc_qc - apply quality control tests to observations, comparisons to the first-guess, comparisons with neighboring observations, vertical consistency checks, assign quality control flags

proc_oa - perform objective analysis independently on each level and each variable

proc_final_analysis - output single time period of data in RAWINS format